Parallelized Quantum Monte Carlo Algorithm with Nonlocal Worm Updates

Masaki-Kato, Akiko; Suzuki, Takafumi; Harada, Kenji; Todo, Synge; Kawashima, Naoki

Parallelized Quantum Monte Carlo Algorithm with Nonlocal Worm Updates

Akiko Masaki-Kato,[1] Takafumi Suzuki,[2] Kenji Harada,[3] Synge Todo,[1,4] and Naoki Kawashima[1]
[1]*Institute for Solid State Physics, University of Tokyo, Chiba, Japan 277-8581*
[2]*Graduate School of Engineering, University of Hyogo, Himeji, Japan 671-2280*
[3]*Graduate School of Informatics, Kyoto University, Kyoto, Japan 615-8063*
[4]*Department of Physics, University of Tokyo, Tokyo, Japan 113-0033*

(Received 17 October 2013; revised manuscript received 10 February 2014; published 10 April 2014)

Based on the worm algorithm in the path-integral representation, we propose a general quantum Monte Carlo algorithm suitable for parallelizing on a distributed-memory computer by domain decomposition. Of particular importance is its application to large lattice systems of bosons and spins. A large number of worms are introduced and its population is controlled by a fictitious transverse field. For a benchmark, we study the size dependence of the Bose-condensation order parameter of the hard-core Bose-Hubbard model with $L \times L \times \beta t = 10240 \times 10240 \times 16$, using 3200 computing cores, which shows good parallelization efficiency.

DOI: 10.1103/PhysRevLett.112.140603 PACS numbers: 05.10.Ln, 02.70.Ss, 67.85.-d

In various numerical methods for studying quantum many body systems, the quantum Monte Carlo (QMC) method, in particular the worldline Monte Carlo method based on the Feynman path integral [1], is often used as one of the standard techniques because of its broad applicability and exactness (apart from the controllable statistical uncertainty). Among its successful applications, most notable are superfluidity in a continuous space [2,3], the Haldane gap in the spin-1 antiferromagnetic Heisenberg chain [4,5], and the BCS-BEC crossover [6]. The QMC method has become more useful due to developments in both algorithms and machines. While global update algorithms, such as loop [7] and worm updates [8], are crucial in taming the QMC methods' inherent problem, i.e., the critical slowing-down, the increase in computers' power following the Moore's law has been pushing up the attainable computation scale. However, it is far from trivial to design the latest algorithms to benefit from the latest machines, since the recent trend in supercomputer hardware is "from more clocks to more cores"; e.g., all top places in the TOP500 ranking based on LINPACK scores are occupied by machines with a huge number of processing cores [9]. As for the loop update algorithm, there is an efficient parallelization, such as the ALPS/LOOPER [10] code, which now makes it possible to clarify quantum critical phenomena with a large characteristic length scale [11]. Unfortunately, the loop update algorithm requires rather stringent conditions about the problems to be studied; it is well known that the algorithm does not work for antiferromagnetic spin systems with external field, nor for bosonic systems with repulsive interactions. In contrast, the worm algorithm enjoys a broader range of applicability [12]. However, the parallelization of the worm algorithm is not straightforward. The reason is simply that the worldline configuration is updated by a single-point object, namely, the worm. This fact makes the whole algorithm event-driven, hard to parallelize. For these reasons, the parallelization of the worm algorithm has been a major challenge from a technical point of view.

In this Letter we present a parallelized multiple-worm algorithm (PMWA) for QMC simulations. A PMWA is a generalization of the worm algorithm and it removes the intrinsic drawback due to the serial-operation nature by introducing a large number of worms. With many worms distributed over the system, it is possible to decompose the whole space-time into many domains, each being assigned to a processor. The neighboring processors send and receive updated configurations on their boundaries, once in every few Monte Carlo (MC) steps. Therefore, the time required for communication can be negligible for sufficiently large systems. Moreover, with a PMWA we can measure an arbitrary n-point Green function which is difficult in conventional worm-type algorithms when $n \geq 4$.

The algorithm described in what follows is based on the directed-loop implementation of the worm algorithm (DLA) [13,14] that samples from the distribution $W(\{\psi_k\}) = \lim_{N_\tau \to \infty} \prod_{k=1}^{N_\tau} \langle \psi_{k+1} | 1 - \Delta\tau \mathcal{H}_\eta | \psi_k \rangle$, where $\Delta\tau = \beta / N_\tau$, $|\psi_k\rangle$ is a basis vector in some complete orthonormal basis set, and $\mathcal{H}_\eta = \mathcal{H} - \eta Q$ is the Hamiltonian with a fictitious source term ηQ that generates discontinuities of worldlines, namely "worms." A configuration in DLA is characterized by a graph, edges and vertices, and state variables defined on edges in the graph. While a vertex is represented by a point in the standard graph theoretical convention, in the literature of the QMC method for lattice systems it is usually represented by a short horizontal line connecting four vertical segments (edges) as in Fig. 1. A vertex at which the local state changes is called a "kink." The update procedure of the conventional DLA consists of two phases; the worm phase

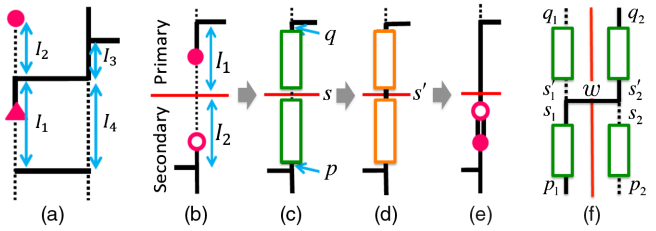

FIG. 1 (color online). A vertex, its four legs, and worms (a), and creation or annihilation of worms on the temporal (b),(c),(d),(e) and on the spatial domain boundary (f). Circles and the triangle are worms whereas horizontal lines are vertices. (b) The configuration before the boundary update. (The red horizontal line is the temporal domain boundary.) (c) The initial intermediate state is $|s\rangle$. (d) The intermediate state is updated. (e) The final configuration compatible to the new intermediate state is generated. (f) The vertex w on the spatial domain boundary, the red vertical line.

in which the motion of the worm causes changes in the state variables, and the vertex phase in which vertices are redistributed. See Refs. [13,14] for details of these updates. While the vertex phase in the new algorithm is just the same as the conventional DLA, the worm phase must be modified. In contrast to the conventional DLA, we let the worms proliferate or decrease freely according to the weight controlled by the parameter η. In conventional DLA, therefore, we "wait" for the worms disappear to measure the observables. (As we see below, configurations with worms are also useful in measuring off-diagonal quantities, i.e., "G-sector measurements" discussed in Refs. [8,15].) In the present algorithm, we estimate them instead by extrapolation to the $\eta = 0$ limit. Corresponding to this modification, the worm update is modified in two ways: worms are created and annihilated at many places at the same time, and we introduce a special update procedure for the region near the boundaries. As a result, the worm phase in the new algorithm consists of three steps: worm creation and annihilation, worm scattering, and a domain-boundary update. The last step is necessary only for parallelization, and is not used when the program runs on a serial machine.

Worm creation and annihilation.—Now we consider how to assign worms on an edge or an interval I separated by two vertices. Generally, the worm-generating operator is defined as $Q_i = \sum_{i,\alpha} Q_{i,\alpha}$ with $Q_{i,\alpha}$ being some local operator and i and α specifying the spatial position and the type of the operator, respectively. (For example, for Bose-Hubbard model $Q_i = \sum_{\alpha=1,2} b_{i,\alpha}$ with the boson annihilation operator $b_{i,1} = b_i$ and the creation operator $b_{i,2} = b_i^\dagger$ at the site i.) As is the case of the graph representation of the $\mathcal{H}$ term, the probability of having n worms in I for a specified sequence of αs, $(\alpha_1, \alpha_2, \ldots, \alpha_n)$ is given by $P_n^{q,p}(I, \{\alpha_k\}) = ((I\eta)^n/n!)\langle q|Q_{i,\alpha_n}\ldots Q_{i,\alpha_1}|p\rangle/f_{qp}(I)$, where $|p\rangle$ and $|q\rangle$ are the initial and the final state of I, respectively, and $f_{qp}(I) \equiv \langle q|e^{I\eta Q_i}|p\rangle$. By taking the summation over all possible sequences, we obtain the probability of choosing n as the number of worms:

$$P_n^{q,p}(I) = \frac{(I\eta)^n}{n!}\frac{\langle q|Q_i^n|p\rangle}{f_{qp}(I)}. \tag{1}$$

Once we have chosen an integer n with this probability, we then choose a sequence of n worms (or αs) with the probability $\langle q|Q_{i,\alpha_n}\ldots Q_{i,\alpha_1}|p\rangle/\langle q|Q_i^n|p\rangle$. After having chosen n and the sequence the n operators in this way, we choose n imaginary times uniform randomly in I and place the n worms there according to the sequence selected above.

While the present algorithm is quite general, let us consider the hard-core Bose-Hubbard model for making the discussion concrete, for which the algorithm becomes simple. In this case, Eq. (1) is nonvanishing only if n is even for $|q\rangle = |p\rangle$ or n is odd for $|q\rangle \neq |p\rangle$, and in either case the sequence that has nonvanishing weight is unique, alternating between b and $b^\dagger$. We here introduce a variable σ, which specifies the "parity" of the number of worms in an interval I; $\sigma = 0$ when $|q\rangle = |p\rangle$ and $\sigma = 1$ when $|q\rangle \neq |p\rangle$. For each parity, the probability, Eq. (1), becomes the following simple form analogous to the Poisson distribution, $P_n^\sigma(I) = ((I\eta)^n/n!)(1/f_\sigma(I))$, $\{n \in \mathbb{N} | n \bmod 2 = \sigma\}$. Here $f_\sigma(I) = \cosh(I\eta)$ for $\sigma = 0$ and $\sinh(I\eta)$ for $\sigma = 1$.

Worm scattering.—Now we consider how we let the worms move around. Note that every worm has the direction, up or down, and according to the nearest object in this direction, different action should be taken. If it is another worm, then we simply change the direction of the worm and do not change its location. If it is a vertex, we let the worm scatter there. Below we discuss how this scattering procedure should be done.

Suppose that a worm is on the ith leg of the vertex. Here a leg is an interval delimited by the vertex in question on one end, and by another vertex or another worm on the other. Then, with probability $P_{\text{enter}} \equiv L_{\min}/L_i$, we let it enter the vertex, where $L_{\min}$ is the length of the shortest of the four legs connected to the vertex [Fig. 1(a)]. Otherwise, we let it turn around without changing its position. If it enters the vertex, it chooses the out-going leg j with probability $P_{\text{scatter}} = w_{ji}/w_i$, where w_i is the weight of the state with the worm on the ith leg. This is the usual scattering probability in DLA. Here, w_{ji} satisfies two equations, $w_{ij} = w_{ji}$ and $w_i = \sum_l w_{li}$, where l runs over all leg indices. Finally, the imaginary time of the worm is chosen uniform randomly in the interval L_j. These procedures define the following transition probability:

$$p_{i\to j} = \frac{L_{\min}}{L_i}\frac{w_{ji}}{w_i}\frac{\Delta\tau}{L_j}. \tag{2}$$

It is obvious that this transition probability satisfies the detailed-balance condition (to be more precise, the time-reversal symmetry condition in the present case),

$p_{i\to j}w_i = p_{j\to i}w_j$. The number of worm scatterings in a MC step is chosen so that every part of the space-time is updated roughly once on average.

Boundary-configuration update.—In the parallelized calculation, we decompose the whole space-time into multiple domains. Then, there are two special cases in the worm scattering discussed above; the case where the worm tries to enter a vertex connecting two domains (spatial domain boundary), and the case where the worm tries to go out of the current domain and enters another (temporal domain boundary). In these two cases, the worm is bounced by the vertex or the boundary with probability one. This treatment obviously satisfies the detailed-balance condition, but it breaks the ergodicity. In order to recover the ergodicity, we carry out the special update procedure described below in the region near the boundary at every MC step.

Figures 1(b)–1(e) show the update procedure of a temporal boundary that has two "legs," I_1 and I_2 [Fig. 1(b)], ending with states $|q\rangle$ and $|p\rangle$, respectively. We choose the processor taking care of the upper domain as the "primary" and let it execute operations for updating the pair. The current local state just at the boundary is $|s\rangle$ [Fig. 1(c)]. Then, the new state $|s'\rangle$ is chosen with the probability,

$$P^{s'}_{\mathrm{dom}} = \frac{f_{qs'}(I_1) f_{s'p}(I_2)}{f_{qp}(I)} \tag{3}$$

[Fig. 1(d)]. Once $|s'\rangle$ has been chosen, we can regenerate all worms in I_1 and I_2 with Eq. (1) as discussed previously [see Fig. 1(e)]. For hard-core bosons, for example, Eq. (3) is explicitly rewritten as $P^{\sigma'_1,\sigma'_2}_{\mathrm{dom}} = (1+\tanh^{S(\sigma'_1)}(I_1\eta)\tanh^{S(\sigma'_2)}(I_2\eta))^{-1}$, where σ_1 and σ_2 are the parities of I_1 and I_2, respectively, and $S(0)=1$ and $S(1)=-1$.

The update procedure of the interdomain vertex is similar to that of the temporal boundary discussed above, although there are four intervals involved in this case instead of two. Suppose we have a vertex with four legs bounded with the ending states $|p_1\rangle$ and $|p_2\rangle$ below the vertex, and $|q_1\rangle$ and $|q_2\rangle$ above the vertex. Now, the new state variables s_1, s_2, s'_1, s'_2 at the roots of the four legs as shown in Fig. 1(f) are stochastically selected according to the product of the vertex weight w and the leg weight f,

$$W^{p_1p_2s_1s_2}_{s'_1s'_2q_1q_2} = f_{q_1s'_1}(I_3) f_{q_2s'_2}(I_4) w^{s_1s_2}_{s'_1s'_2} f_{s_1p_1}(I_1) f_{s_2p_2}(I_2), \tag{4}$$

where $w^{s_1s_2}_{s'_1s'_2}$ is $\langle s'_1, s'_2|H_{\mathrm{pair}}|s_1, s_2\rangle$, previously referred to as w_i in Eq. (2) with i representing the set root states s_1, s_2, s'_1, s'_2. Once the new root states have been selected, the rest of the task is the same as the temporal boundary update; i.e., we regenerate worms on the four legs. These tasks are carried out by the primary processor that takes care of the "left"-hand side of the vertex.

Pseudo code.—We summarize all of the procedure described above in the form of a pseudo code. The task of a processor ν in a MC step is as follows.

(Step 1) Send to and receive from the neighboring processor the ending states of the intervals on the temporal boundary. For each one of the intervals of which ν is primary, select the intermediate state stochastically with the probability Eq. (3). Send and receive the updated intermediate states.

(Step 2) Send to and receive from the neighboring processor the ending states of the legs of the vertices on the spatial boundary. For each one of the vertex of which ν is primary, select the states at the roots of the legs stochastically with the weight Eq. (4). Send and receive the updated root states.

(Step 3) As in the conventional DLA, erase all vertices without a kink on it, and place new vertices with the density proportional to the corresponding diagonal matrix element of the Hamiltonian.

(Step 4) For each interval I delimited by the vertices or the domain boundaries, erase all the worms, generate an integer n with the probability Eq. (1), generate a sequence of n operators, and place them uniform randomly on I. Also choose the direction of each worm with probability $1/2$.

(Step 5) For every worm, if the nearest object ahead is a vertex that is not on a boundary, let it enter the vertex with the probability P_{enter}, let the worm scatter there with the probability P_{scatter}, and choose the imaginary time uniform randomly on the final leg. Otherwise, reverse its direction without changing its position.

(Step 6) Repeat Step 5 $N_{\mathrm{cycle}}-1$ more times, and perform measurements. This concludes the MC step.

Benchmark.—We apply the algorithm to the hard-core Bose-Hubbard model in the square lattice. The model we consider here is defined as

$$\mathcal{H} = -t\sum_{\langle i,j\rangle} b_i^\dagger b_j + V\sum_i n_i n_j - \mu\sum_i (n_i + n_j), \tag{5}$$

where μ is the chemical potential and V denotes the nearest-neighbor interaction. In the PMWA, we simulate the Hamiltonian $\mathcal{H}_\eta$ to generate multiple worms, then we extrapolate the QMC results to the $\eta=0$ limit. The extrapolation rule is given by the expansion of the physical quantity in a power series of η where η is small. For example, the coefficient of the first order term of the energy is as follows:

$$\left.\frac{\partial\langle\mathcal{H}\rangle_\eta}{\partial\eta}\right|_{\eta\to 0} = -\langle Q\rangle_0 + \beta\langle\mathcal{H}Q\rangle_0 - \beta\langle\mathcal{H}\rangle_0\langle Q\rangle_0, \tag{6}$$

where $\langle\cdots\rangle_0$ is the mean value with respect to the non-perturbed Hamiltonian Eq. (5). When we choose Q to be a measure of the spontaneous symmetry breaking, as we do below, the right-hand side of Eq. (6) is always 0 for a finite

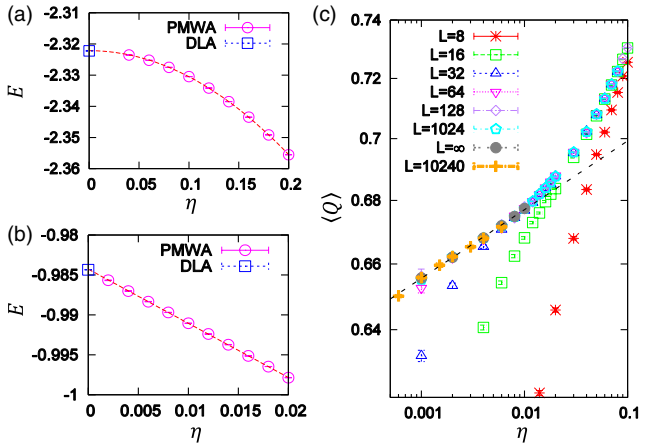

FIG. 2 (color online). Energy E and order-parameter Q as a function of the source field η. (a) and (b): Energy at fixed system size, temperature, and repulsive interaction ($L = 128$, $\beta t = 16$, $V/t = 3.0$). The chemical potential is $\mu/t = 4.2$ (CS phase) for (a), and $\mu/t = 1.2$ (SF phase) for (b). The dashed curves represent a quadratic fitting for (a) and a linear fitting for (b). (c): Double logarithmic plot of $\langle Q \rangle$ at $\beta t = 16$, $\mu/t = 1.2$ and $V/t = 3.0$. The dashed line is the power-law fitting with $L = \infty$ data.

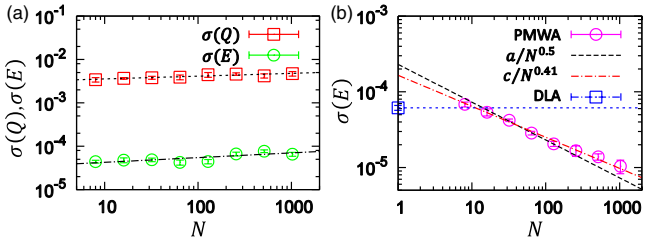

FIG. 3 (color online). The estimated standard deviation (error) as functions of the number of domains N in a SF state ($\mu/t = 1.2$, $V/t = 3.0$). (a) With fixed number of MC sweeps. ($L = \beta t = 128$, $\eta = 0.002$), (b) With fixed wall-clock time of 10^4 seconds (not including thermalization). ($L = 256$ and $\beta t = 16$, $\eta = 0.004$)

system, making the $\mathcal{O}(\eta)$ term in $\langle \mathcal{H} \rangle$ vanish. This result leads us to quadratic extrapolation $-2.32216(4)$, which shows good agreement with the conventional DLA result $-2.32222(2)$ in the checkerboard solid (CS) phase [Fig. 2(a)]. In contrast, in the superfluid (SF) phase $\langle Q \rangle$ is finite in the thermodynamic limit at zero temperature. Even for finite systems at finite temperature, the deviation from the thermodynamic behavior at $T = 0$ appears only in very small η and is practically not observed in large systems for which parallelization is necessary. It allows us to extrapolate the energy linearly at low temperatures as we see in Fig. 2(b) in which values are $-0.98431(1)$ by the PMWA and $-0.98434(2)$ by DLA. By closer inspection, however, we can also estimate the continuously varying scaling exponent, characteristic of the two-dimensional systems at finite temperature. Below we demonstrate that the present method can produce the off-diagonal order parameter, namely, the Bose-Einstein condensate (BEC) order parameter $\langle Q \rangle$. The procedure for measuring this quantity and an arbitrary multipoint Green function, i.e., G-sector measurements, is shown in the Supplemental Material [16]. Figure 2(c) shows the numerical results for systems ranging from $L = 8$ to 10240 at fixed $\beta t = 16$, which is much larger than a single processor can accommodate. We also present the result of extrapolation to the infinite L limit for each value of η based on the results of $L \leq 1024$. The $L = 10240$ results calculated by using 3200 CPU cores agree well with the extrapolation. The dashed line is the power law fitting from which we can read the magnetization critical exponent $1/\delta$.

It is in general possible that the domain boundaries hinder the propagation of the locally equilibrated region, and cause a slowing-down. In order to see the seriousness of this effect, we estimated the standard error, i.e., 1 standard deviation of the expected distribution, of the mean values of the order parameter $\sigma(Q)$ and that of the energy $\sigma(E)$ in SF phase as a function of the number of domains N in Fig. 3(a). We measured the quantities at every MC sweep, and the averages are taken over the same number of MC sweeps. Though the number of worms decrease with increasing N, in all of our presented simulation the average number of worms in each domain is from $O(10)$ to $O(100)$ for used η, and the probability of finding an empty domain is very low. We find a weak N dependence of σ, which is empirically described as $\sigma \propto N^{0.09}$. We here emphasize that the PMWA is also efficient from the technical point of view; i.e., each processor has to communicate only with its neighbors, and the amount of transmitted information is proportional to the area of only the interface. This property is manifested in the good parallelization performance of our algorithm and code. For the so-called "weak scaling" performance, with the system size being proportional to the number of processors, see the Supplemental Material [16]. Figure 3(b) suggests good "strong scaling" performance, with the fixed system size and increasing number of processors. Specifically, it shows the standard error as a function of N with both the system size and the elapsed time ("wall-clock" time) fixed. It plainly shows that we can achieve higher accuracy by employing more processors. The N dependence of the error is described (again empirically) by $N^{-0.41}$ which is slightly worse than the ideal dependence $N^{-0.5}$. The small difference in the exponent 0.09 comes from the slight increase observed in Fig. 3(a).

We have presented a PMWA, a new parallelizable QMC algorithm, which can treat extremely large systems. We have applied it to hard-core bosons and observed high parallelization efficiency. The multibody correlation function should be computed relatively easily in the new algorithm. In addition, the PMWA can be extended in several ways. For example, "on-the-fly" vertex generation [17,18], in which vertices are generated only in the immediate vicinity of the worms, is possible. Another

extension may be the "wormless" algorithm. Obviously, the boundary update in terms of the parity of the number of worms rather than worms themselves can be used also for updating bulk regions. By doing so, we can altogether get rid of worms. These extensions will be discussed elsewhere [19]. The source code of our program will be released in Ref. [20] in the near future.

We would like to thank H. Matsuo, H. Watanabe, T. Okubo, R. Igarashi, and T. Sakashita for many helpful comments. This work was supported by CMSI/SPIRE, the HPCI System Research project (hp130007), and Grants-in-Aid for Scientific Research No. 25287097. Computations were performed on computers at the Information Technology Center of the University of the Tokyo, and at Supercomputer Center, ISSP.

[1] M. Suzuki, Prog. Theor. Phys. **56**, 1454 (1976); M. Suzuki, S. Miyashita, and A. Kuroda, Prog. Theor. Phys. **58**, 1377 (1977).
[2] D. M. Ceperley, Rev. Mod. Phys. **67**, 279 (1995).
[3] P. Corboz, M. Boninsegni, L. Pollet, and M. Troyer, Phys. Rev. B **78**, 245414 (2008).
[4] F. D. M. Haldane, Phys. Rev. Lett. **50**, 1153 (1983).
[5] M. P. Nightingale and H. W. J. Blöte, Phys. Rev. B **33**, 659 (1986).
[6] K. Van Houcke, F. Werner, E. Kozik, N. Prokof'ev, B. Svistunov, M. J. H. Ku, A. T. Sommer, L. W. Cheuk, A. Schirotzek, and M. W. Zwierlein, Nat. Phys. **8** 366 (2012).
[7] H. G. Evertz, G. Lana, and M. Marcu, Phys. Rev. Lett. **70**, 875 (1993).
[8] N. Prokof'ev, B. Svistunov, and I. Tupitsyn, Sov. Phys. JETP **87**, 310 (1998).
[9] http://www.top500.org/.
[10] B. Bauer *et al.*, J. Stat. Mech. (2011) P05001.
[11] K. Harada, T. Suzuki, T. Okubo, H. Matsuo, J. Lou, H. Watanabe, S. Todo, and N. Kawashima, Phys. Rev. B **88**, 220408(R) (2013).
[12] S. Trotzky, L. Pollet, F. Gerbier, U. Schnorrberger, I. Bloch, N. V. Prokof'ev, B. Svistunov, and M. Troyer, Nat. Phys. **6**, 998 (2010).
[13] O. F. Syljuåsen and A. W. Sandvik, Phys. Rev. E **66**, 046701 (2002).
[14] N. Kawashima and K. Harada, J. Phys. Soc. Jpn. **73**, 1379 (2004).
[15] A. Dorneich and M. Troyer, Phys. Rev. E **64** 066701 (2001).
[16] See Supplemental Material at http://link.aps.org/supplemental/10.1103/PhysRevLett.112.140603 for the procedure of measuring the BEC order parameter and the multipoint Green function, and for the acceleration efficiency for weak scaling of PMWA.
[17] Y. Kato, T. Suzuki, and N. Kawashima, Phys. Rev. E **75**, 066703 (2007).
[18] L. Pollet, K. V. Houcke, and S. M. A. Rombouts, J. Comput. Phys. **225**, 2249 (2007).
[19] A. Masaki-Kato, T. Suzuki, K. Harada, S. Todo, and N. Kawashima (to be published).
[20] http://ma.cms-initiative.jp/en/application-list/dsqss/dsqss.